Course on Database Design

Carlo Batini

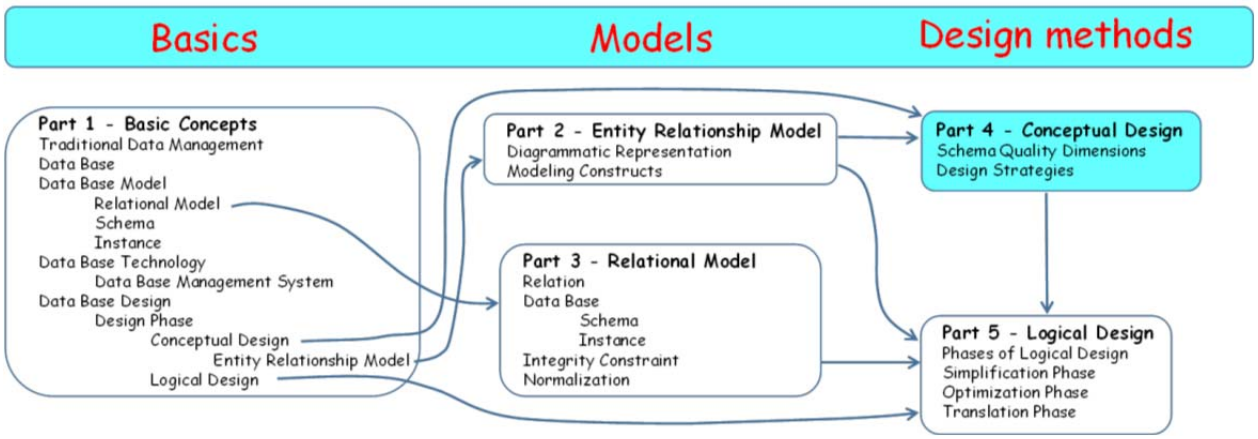
University of Milano Bicocca

Course on Database Design

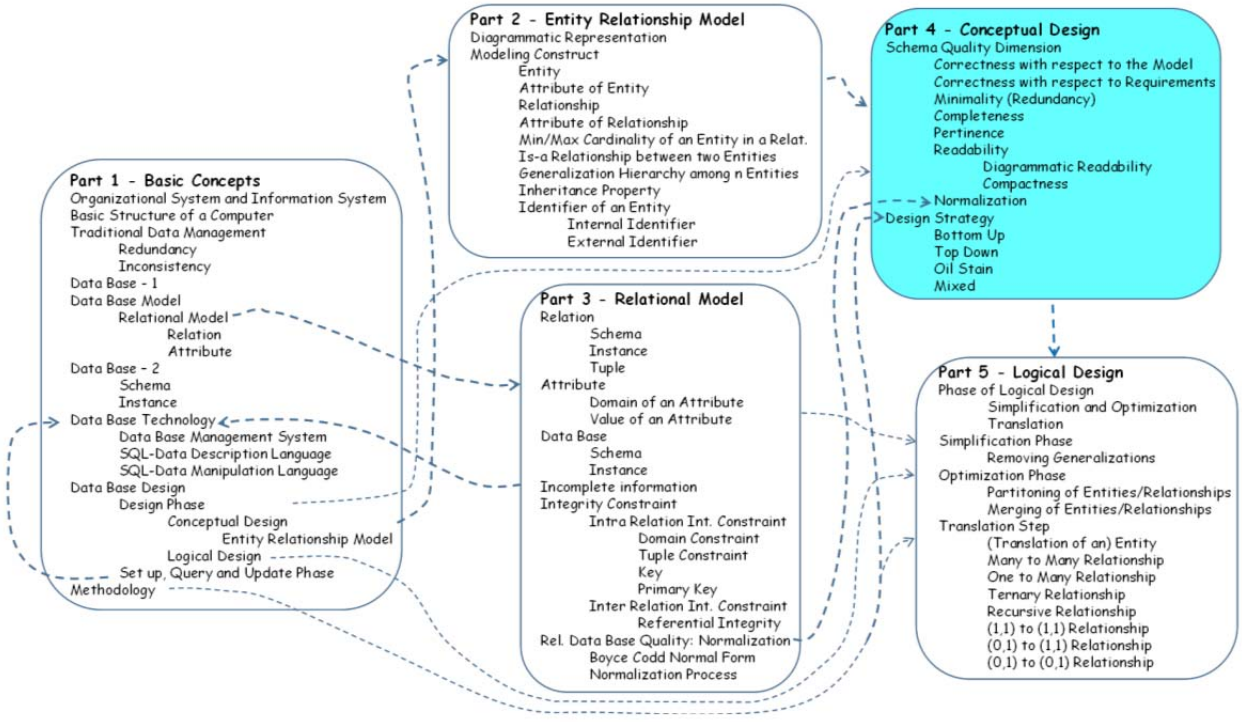
Part 4 – Conceptual Design

Part 4 – Lesson 1 – Introduction to Conceptual Design

We start in Part 4 to examine the issue of design methods, that is twofold, due to the separation between the conceptual phase and the logical phase. The conceptual phase is focused on modeling the reality of interest using modeling constructs that are closer to the user and independent from the model adopted by the DBMS. The logical phase results in a representation of requirements that can be processed by the DBMS.



High level conceptual map



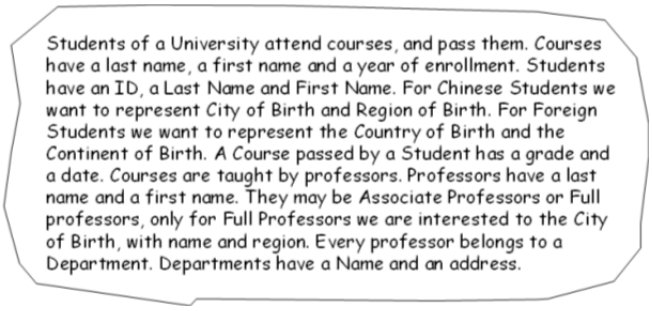
Low level conceptual map

If we look at part 4 in the low level conceptual map, we see that the issues dealt with are essentially two:

a. a discussion on *qualities* of the conceptual schema to be produced by conceptual design, namely "good properties" that the conceptual schema shall respect, and

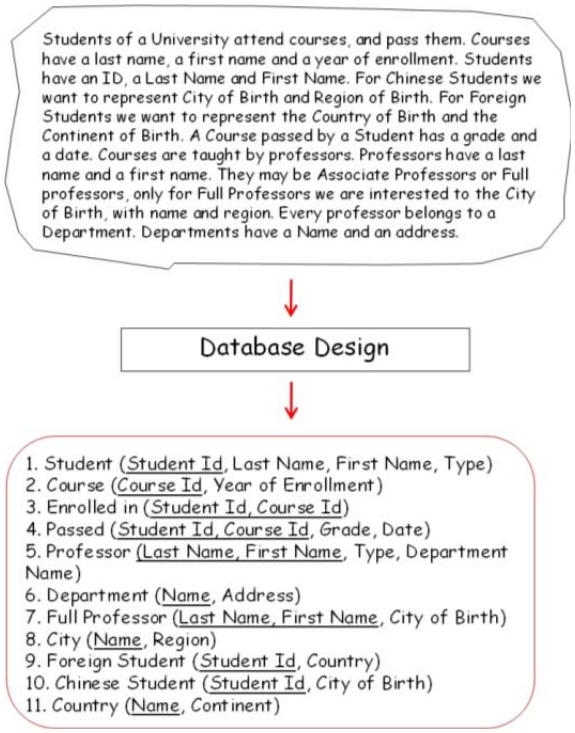
b. an analysis of *strategies* at our disposal for dealing with all the modeling decisions to be taken during design.

To understand better and in more depth the two issues, qualities and strategies, consider requirements in the figure below, more complex than previous requirements I have proposed to you.



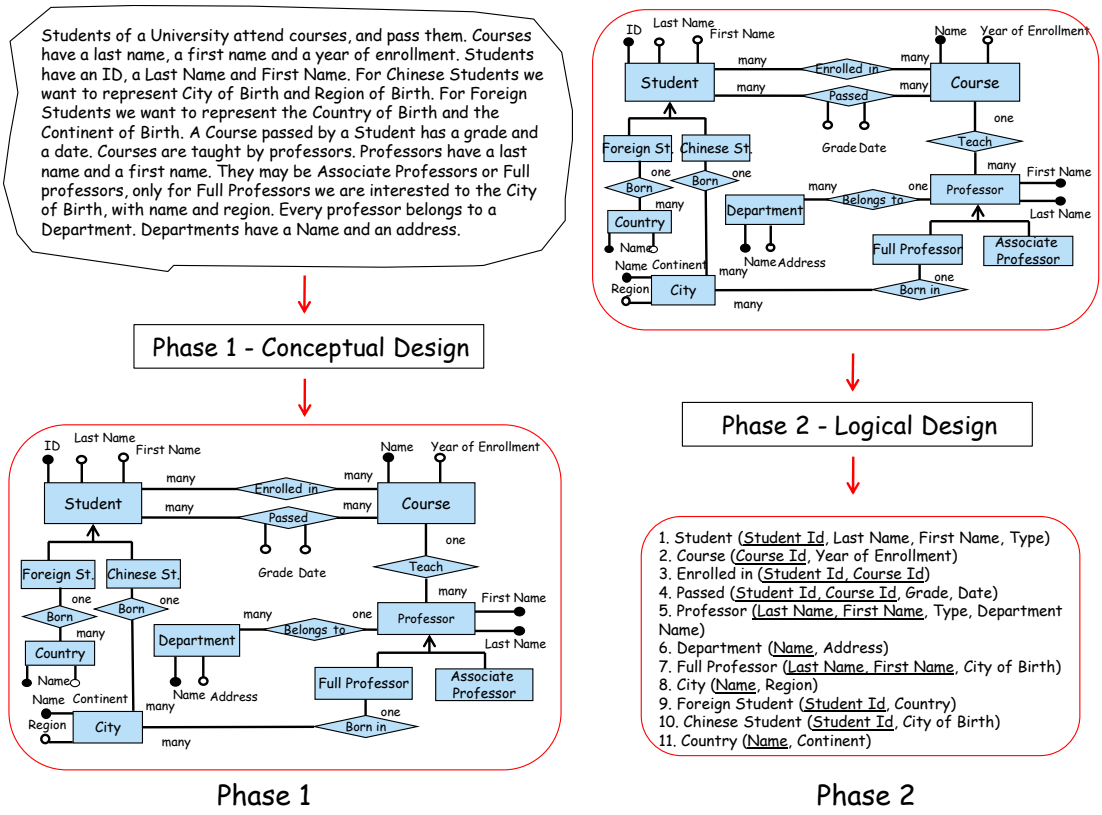
An example of complex requirements

The context is as previously a University, but now concepts involved in requirements are many, such that it is not possible a design process "all at once", as the one depicted in the following figure, that produces directly a logical relational schema.



Design process "all at once", producing directly a logical schema

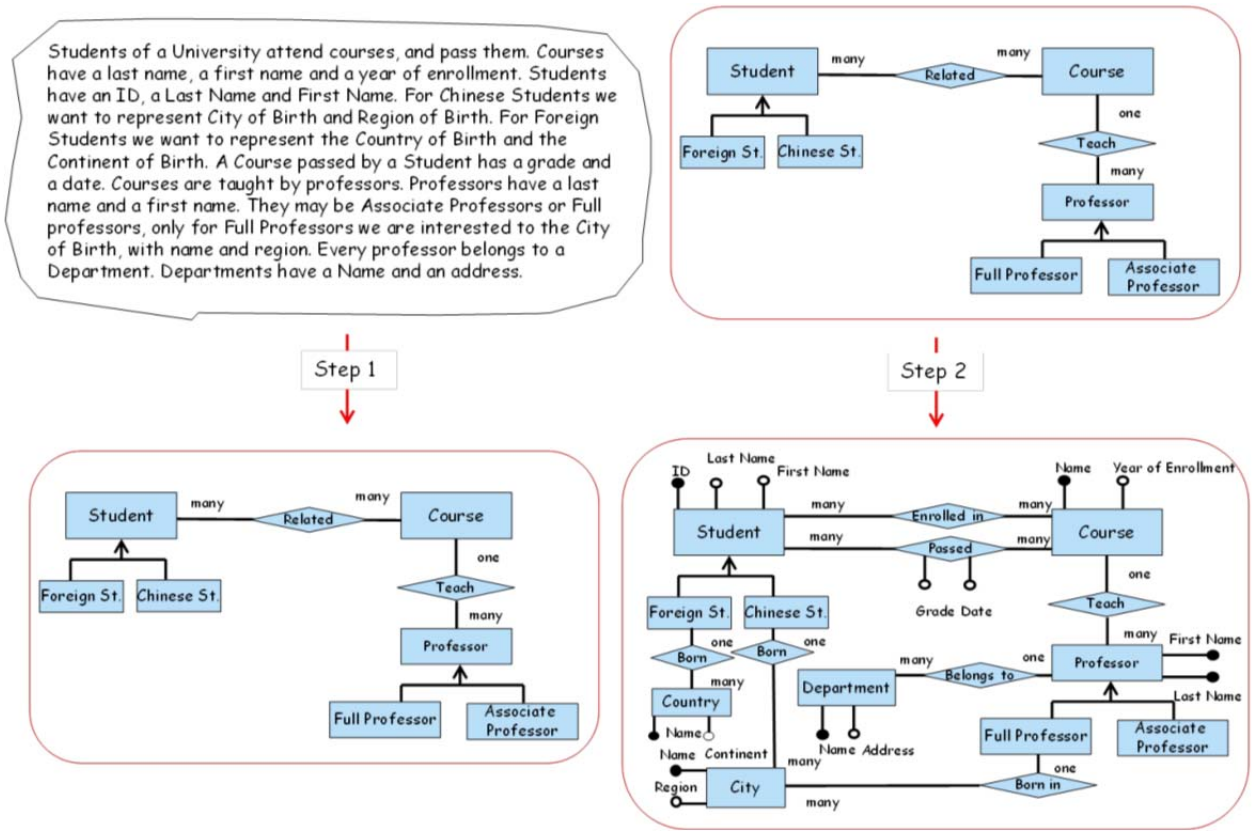
For this reason, a first general strategy we can adopt reduces complexity by decomposing database design in two phases, as in the following figure and as in our course.



Two phases design to reduce complexity

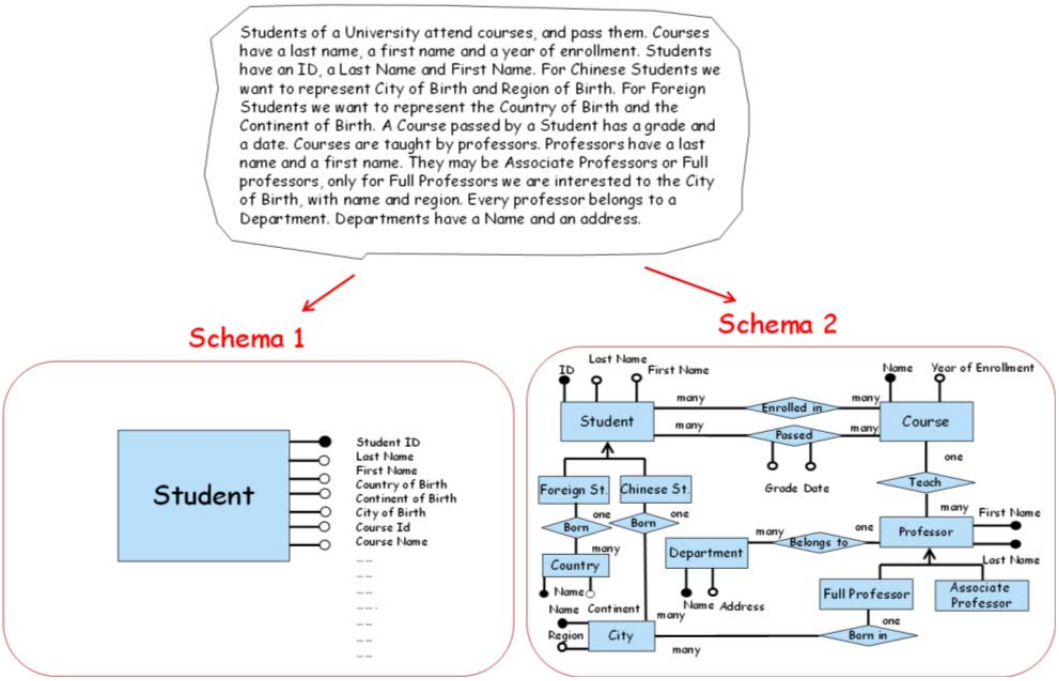
But this is not enough. Also “inside” conceptual design we need a strategy that avoids a one shot process; in other words, we need to establish a coordinated set of steps that we can call *strategy*, namely a high level plan to achieve a goal. As in the following figure, where we first model most important concepts (Step 1) and then we complete the schema with remaining entities and relationships (Step 2).

Notice, as an example, that the relationship between Student and Course is initially seen as a generic relationship, while in the second step it is refined into the two “final” relationships *Ennrolled in* and *Passed*.



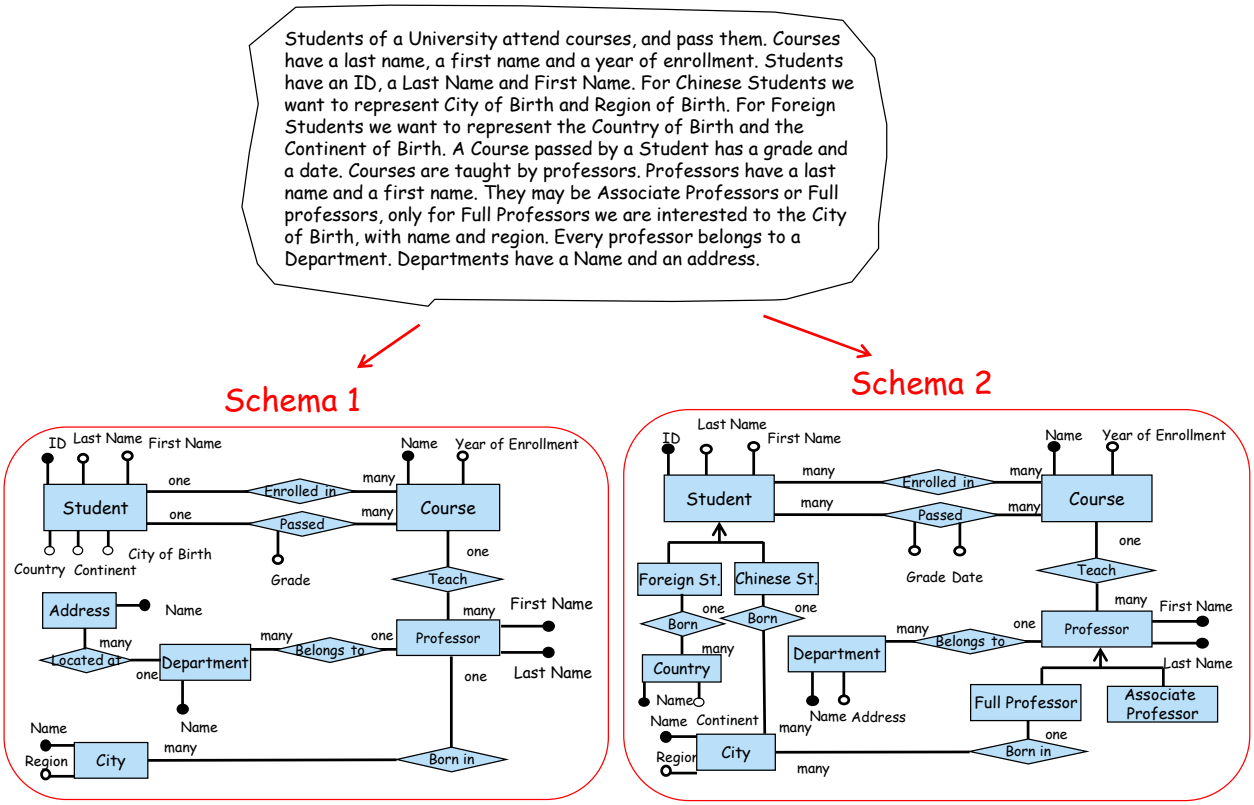
An example of a two step design process that can be adopted in conceptual design

On the other hand, strategies too are not enough. E.g. we can imagine, as in the following figure, that requirements have been modeled with two different design processes, leading to Schema1 and Schema2.



Two very different schemas that can be produced in conceptual design

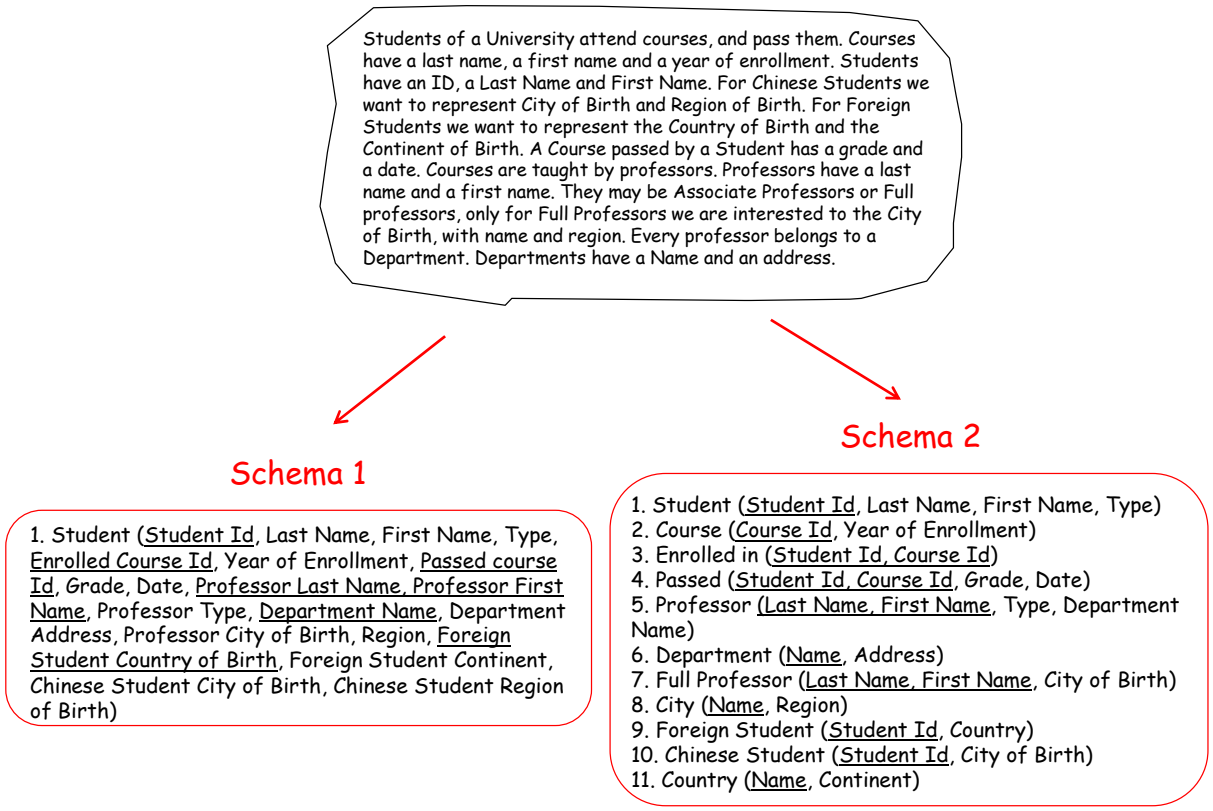
It is intuitive that Schema 1, made of only one big entity whose attributes model all the concepts in requirements, is not a "good" schema. It is intuitive, but it is not easy to formalize suitable quality dimensions that capture the adjective "good". For instance, in the following figure we have two ER schemas that aims to represent all the requirements, and apparently are of similar quality.



Schemas of dissimilar "quality" produced by the same requirements

Looking in more detail, we see that in Schema 1 we missed to represent the date of exam, and, further, we have represented the address with an entity, while, according to the meaning of the concepts of entity and of attribute in the ER model, we may say that address is better represented as an attribute of the entity Department.

As a last example, consider the following figure, that directly produces from the requirements relational schemas without an intermediate conceptual step.



Quality in relational schemas

Also in this case, we say that Schema 2 is of higher quality than Schema 1, but here we already know a powerful quality criterion, the property of Boyce Codd Normal Form, that allows us to say that Schema 2 is in BCNF, while Schema 1 is not. And we know that respecting BCNF leads to relation instances that are not redundant and where each datum is represented only once, so that update anomalies are avoided.

In conclusion, I hope you agree that in the database design activity, we need to examine in more depth two issues,

a. schema quality dimensions, and
b. schema design strategies.

We will address these issues in the following lessons.

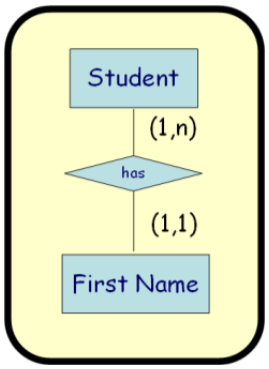
Part 4 – Lesson 2 – First set of schema design qualities

In this lesson and in the following one I will introduce the following schema design qualities:

- Correctness with respect to the model
- Correctness with respect to requirements
- Minimality (and its opposite, redundancy)
- Completeness
- Pertinence
- Diagrammatic Readability
- Compactness
- Normalization (in the Entity Relationship model)

Correctness with respect to the model

Correctness with respect to the model concerns the correct use of the constructs of the model in representing requirements. As an example, assume we want to represent persons and their first names. If we produce the following schema, we may say that the schema is not correct w.r.t. the model, since an entity should be used only when the concept has a unique existence in the real world and has an identifier, so First Name is not an entity.



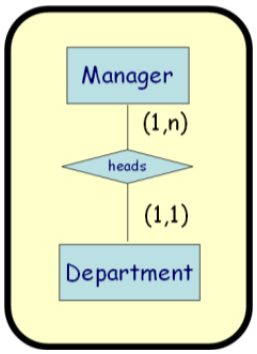
A schema that is not correct w.r.t. the model

Definition - *Correctness w.r.t. the model* means that modeling constructs in the schema are used according to their definition and properties.

Correctness with respect to requirements

Definition - Correctness with respect to requirements concerns the correct representation of the requirements in terms of the model constructs.

Example – Assume that an organization each department is headed by exactly one manager and each manager may head exactly one department.

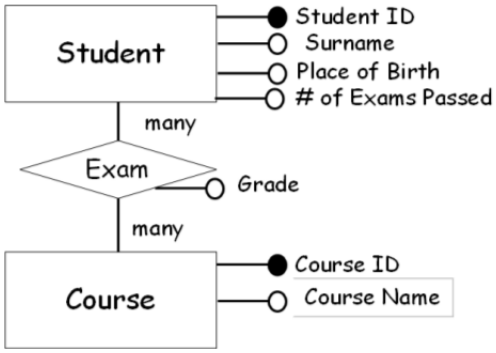


A schema that is not correct w.r.t. requirements

If we represent Manager and Department as entities, the relationship between them (see the schema above) should be one-to-one; in this case, the schema is correct w.r.t. requirements. If we specify a one-to-many relationship, the schema is not correct.

Minimality

Moving to minimality, consider the following schema.



A schema that is not minimal

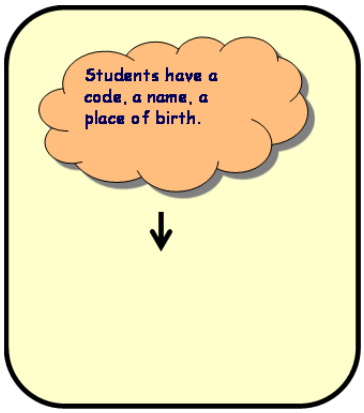
The value of the attribute *# of Exams Passed* can be calculated counting for each student the number of instances of the relationship *Exam* in which the student is involved. So we say that the schema has a redundancy, or that it is not minimal.

Definition - *Minimality* states that redundant concepts, namely concepts that can be expressed in terms of other concepts in the schema, are absent.

I observe that when we find redundancies in the conceptual schema, we don't have necessarily to remove them; as we will see in logical design, a redundancy in the logical schema may lead to execute more efficiently certain queries. Redundancies have to be simply *checked* and put in evidence in conceptual design.

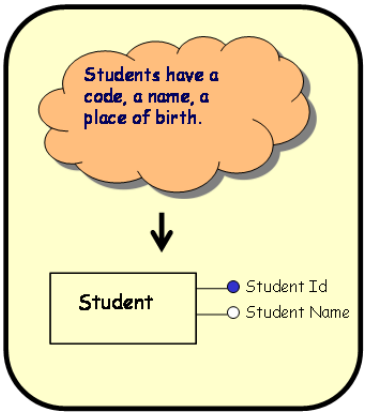
Completeness and pertinence

To introduce completeness, look at the following requirements.



Example of requirements

and assume we model them with an entity and attributes *Student Id* and *Name*. In this case we say that the schema is not complete compared with requirements.

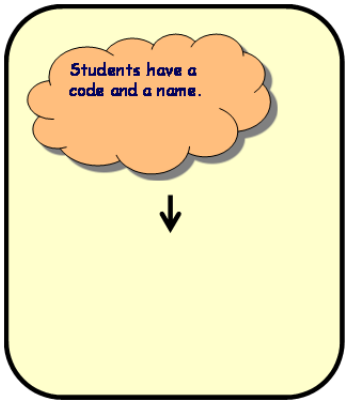


A schema that is not complete

Definition - *Completeness* measures the extent to which a conceptual schema includes all the conceptual elements necessary to meet requirements.

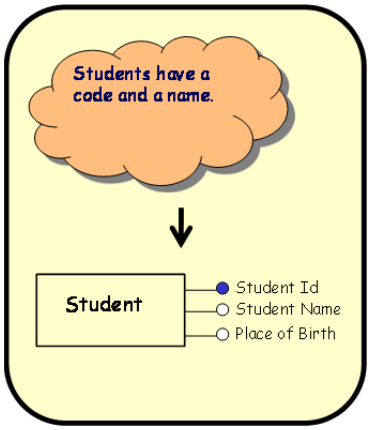
Pertinence

Definition - *Pertinence* is the complementary quality of completeness, and measures how many unnecessary conceptual elements are included in the conceptual schema.



Example of requirements

E.g., if we want to model requirements in the above figure, and we produce an entity with attributes *Student Id*, *Name*, and, say, *Place of Birth*, we have represented a concept, Place of Birth that is not pertinent, as it is not contained in requirements.

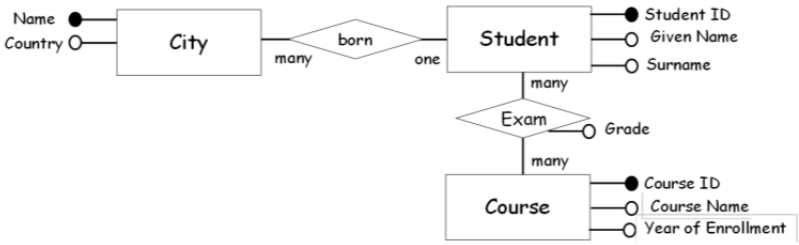


A schema that is not pertinent

How to assess completeness and pertinence

So far in this lesson, we have defined several schema quality dimensions, but we did not provide a precise procedure to measure them. We show now a procedure for completeness and pertinence, using an example of requirements. Assume that requirements and the related schema are as in the following.

Students have Student Id, Given Name, Surname, Date of Birth, City of Birth, Country of Birth. Courses have Course Id, Name, Year of Enrollment, Number of hours. Student pass exams, with Grade and Date.



A schema that should represent the above requirements: which is its level of completeness?

To check the completeness of the schema, we may examine requirements, marking all names and verbs that are present, and check whether there is a corresponding construct in the schema, see the next box.

~~*Students have Student Id, Given Name, Surname,*~~ *Date of Birth,* ~~*City of Birth, Country of Birth.*~~

~~*Courses have Course Id, Name, Year of Enrollment*~~*, Number of hours.* ~~*Student pass exams, with Grade*~~ *and Date.*

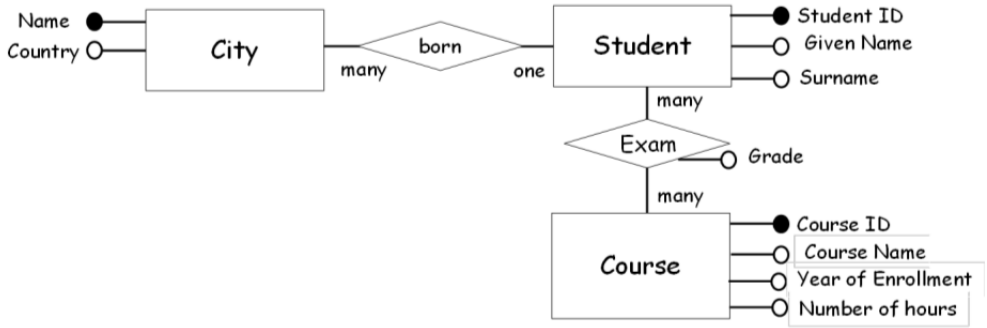
If some terms do not have a correspondence, we say that the schema is incomplete. In our example the schema is not complete, as *Date of Birth*, *Number of hours* and *Date of exam* are not represented in the schema.

To check the pertinence of the schema, we have to follow the complementary path. For each construct in the schema. we have to map the construct with a corresponding term in the requirements; if in some case we do not succeed to map the construct, this means that the schema is not pertinent compared to requirements.

Exercise 4.1 - Assume now requirements are as follows.

Students have Student Id, Given Name, Surname, City of Birth, Country of Birth. Courses have Course Id, Name, Year of Enrollment. Student pass exams, with grade.

and the schema is



Is the schema pertinent?

Answer to Exercise 4.1

Following the procedure we discover that the attribute *Number of hours* does not appear in requirements, so the schema is not pertinent. With this last issue, we have concluded the lesson.

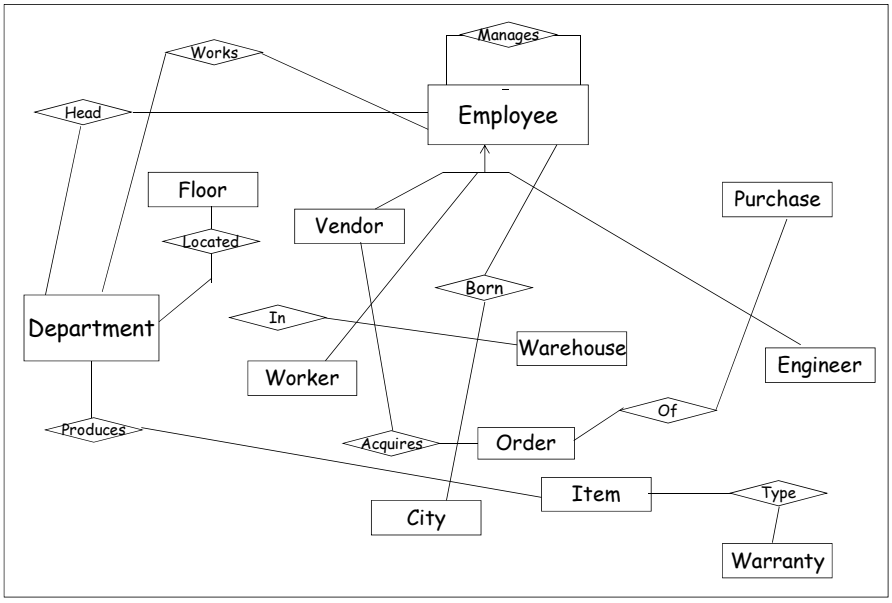
Part 4 – Lesson 3 - Schema Quality dimensions – second part

In this lesson we focus on the remaining schema quality dimensions, namely:

- diagrammatic readability,
- compactness and
- ER schema normalization.

Diagrammatic readability

To introduce diagrammatic readability, look at the following schema.



A schema that is perceived mixed up

In this case I do not provide you the requirements in natural language, and do not provide information to understand in detail the meaning of the constructs in the schema, this is not my goal. I invite you to look at the schema from the point of view of readability, that corresponds to your subjective perception of the clarity with which the diagram is drawn, and the related ease of understanding.

We can agree on the fact that the diagram is badly drawn. It is not so easy to say why we judge the diagram badly drawn. I have tried in the past to list a set of *aesthetic criteria* that formalize the concept of well drawn diagram. The most important, in my opinion, are:

1. Graphic symbols should be embedded in a grid.
2. Crossings between lines should be minimized.
3. Lines should be made of horizontal or vertical segments.

If you are not bored by these aesthetic criteria, I will add more:

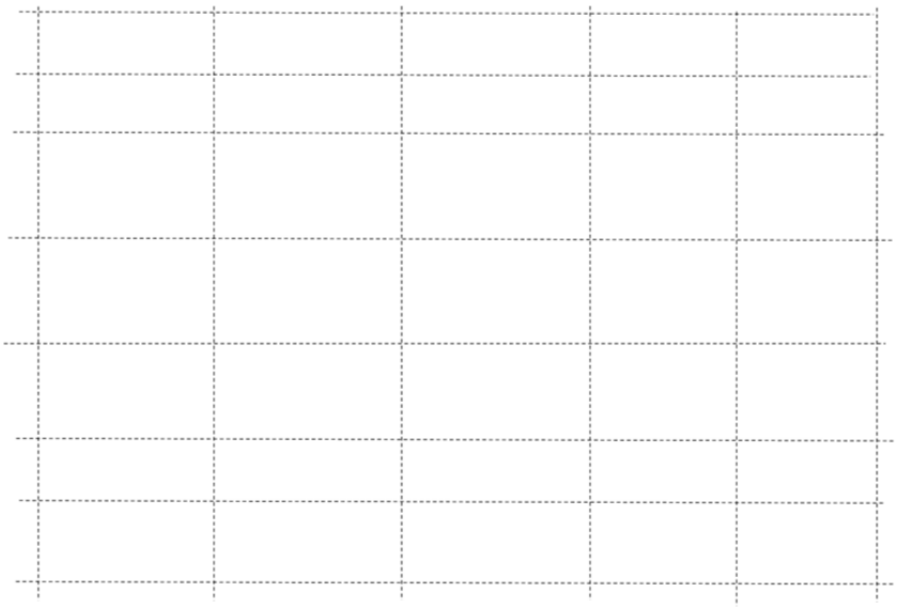
4. The number of bends in lines should be minimized.
5. The total area of the diagram should be minimized.
6. The parent in a generalization hierarchy should be positioned at a higher level in the diagram in respect to children.

7. Children entities in a generalization hierarchy should be symmetrical with respect to the parent entity.

So we can say that *diagrammatic readability* means that the diagram is easily understandable, as several aesthetic criteria are adopted. Let us try now to apply the first criterion to the diagram above:

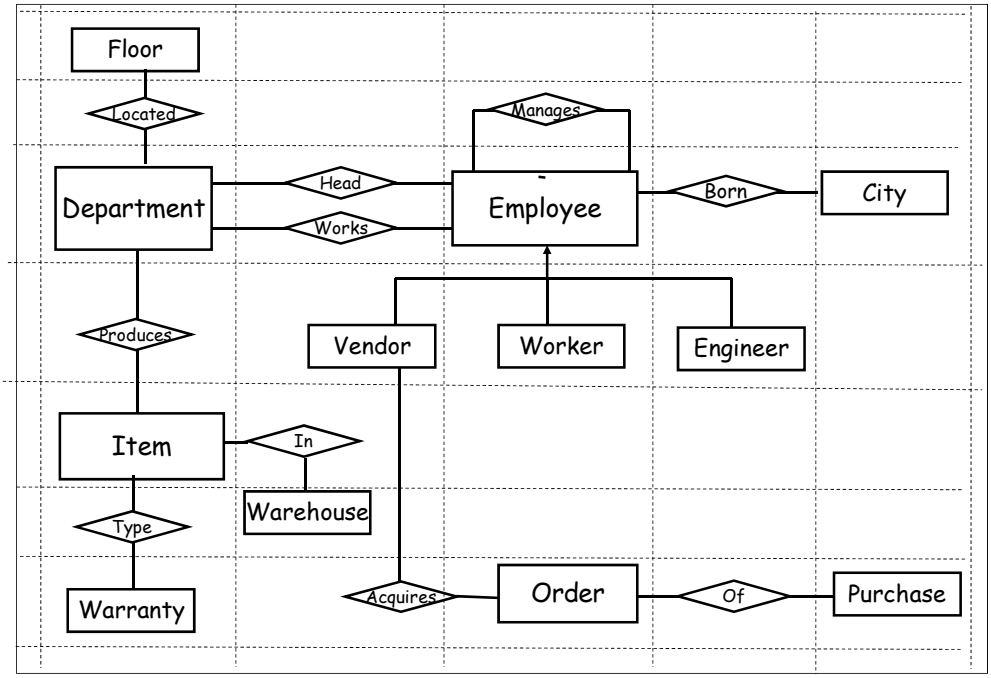
Graphic symbols should be embedded in a grid.

Question 4.2 - To facilitate your job, I will provide you a grid. Try to place graphic symbols in the grid, and then lines, and do not worry if a symbol and a line occupy the same cell in the grid.



Answer to Question 4.2

A good solution is as follows.

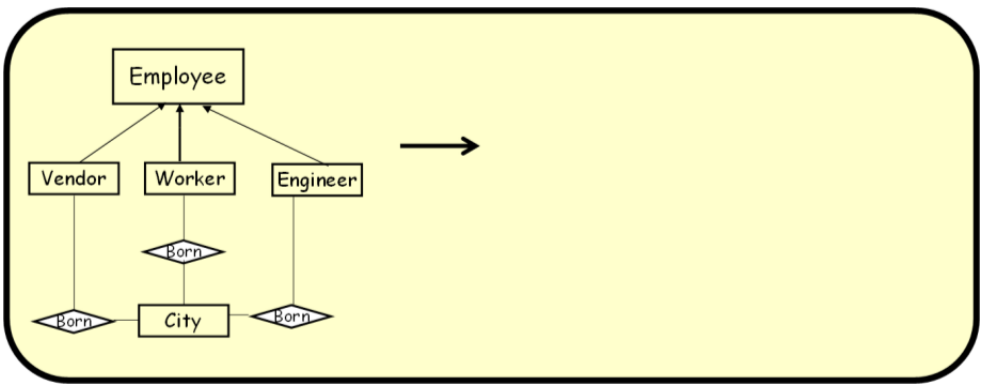


Notice that while applying the embedding in the grid as a sort of side effect, we have also achieved the second and third criterion:

2. Crossings between lines should be minimized.
3. Lines should be made of horizontal or vertical segments.

Compactness

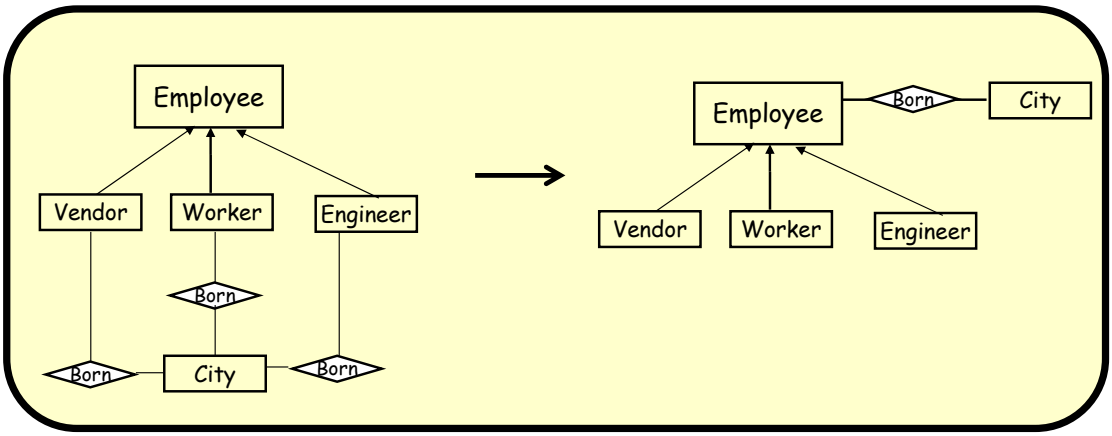
Question 4.3 - To introduce compactness look at the following schema. Try to transform the following schema in such a way that you get an equivalent schema, namely a schema that represents the same requirements, that uses a lower number of constructs.



Try to transform the schema….

Solution to Question 4.3

We can exploit the inheritance property, and refer the entity *City* and the related relationship *Born* directly to the entity *Employee*, leading to the following schema.

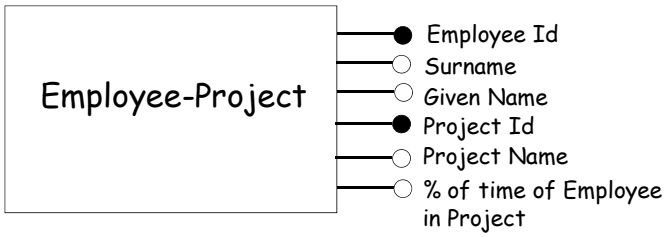


A more compact schema

Definition - *Compactness* of schema representation leads to choose, among the different conceptual schemas that equivalently represent a certain reality, the one or the ones that are more compact, i.e. use few concepts.

Notice that compactness favors readability, since minimizes the cognitive effort to understand the meaning of the schema.

Normalization in the ER model

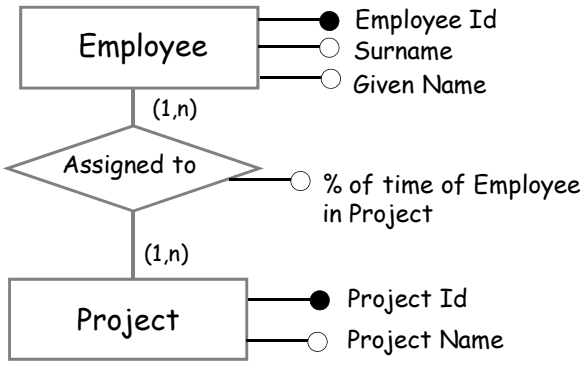


An un-normalized schema in the ER model

Look at the above schema, made of one entity. Let us try to understand the meaning of the entity, and make the opposite exercise compared to all our previous exercises; let us try to express the meaning of the schema by means of a set of sentences in natural language.

First of all, notice the name of the entity that is composed of two terms, Employee and Project. It is the first time we find an entity whose name is made of two terms. Looking at the identifier, again, it is made of two attributes; this is a clue that the they identify two concepts, or two entities, namely Employee and Project. Observing the other attributes we easily conclude that *Surname* and *Given Name* are attributes of entity *Employee*, *Project Name* is an attribute of an entity Project, and *% of time of Employee in Project* is not an attribute of Employee, it is not an

attribute of Project, it is an attribute that can be associated to a relationship between Employee and Project. The above analysis leads us to transform the schema into the following one.



A normalized schema in the ER model

We see from the above example that also in the ER model we can define a property of *normalization*.

Definition - An ER schema is in normal form, or in Boyce Codd Normal form, when every distinct class of objects of the reality is represented by means of a different entity of relationship.

Notice that we could define the concept of normalized schema in the ER model, introducing also in the ER model a concept of functional dependency among attributes of entities and relationships. We prefer to adopt a different, more "qualitative" approach, to confirm the point of view that the ER model is a *conceptual* model, so it is a model whose constructs are closer to the human way of thinking. With this consideration, I have concluded the lesson.

Part 4 – Lesson 4 - Strategies in conceptual design

To explain the issues related to strategies in conceptual design I need animation, and so I decided to use Power Point in the video lesson. In general I do not like Power Point, but in this case it is more effective than One Note, that I adopt in all other lessons. I would like first to discuss the strategies looking at their adoption starting from a same set of requirements. The strategies I will compare are three:

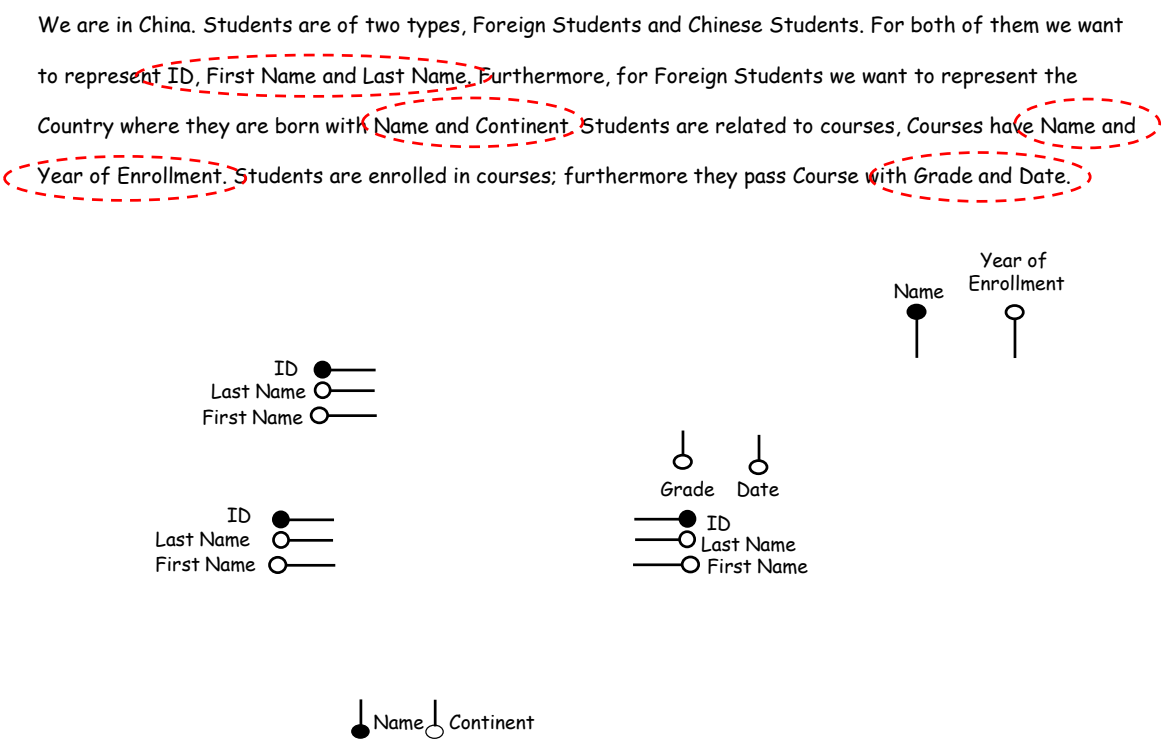
- Bottom up
- Inside outside
- To down.

The requirements are the following.

We are in China. Students are of two types, Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses, Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Courses with Grade and Date.

Bottom up strategy

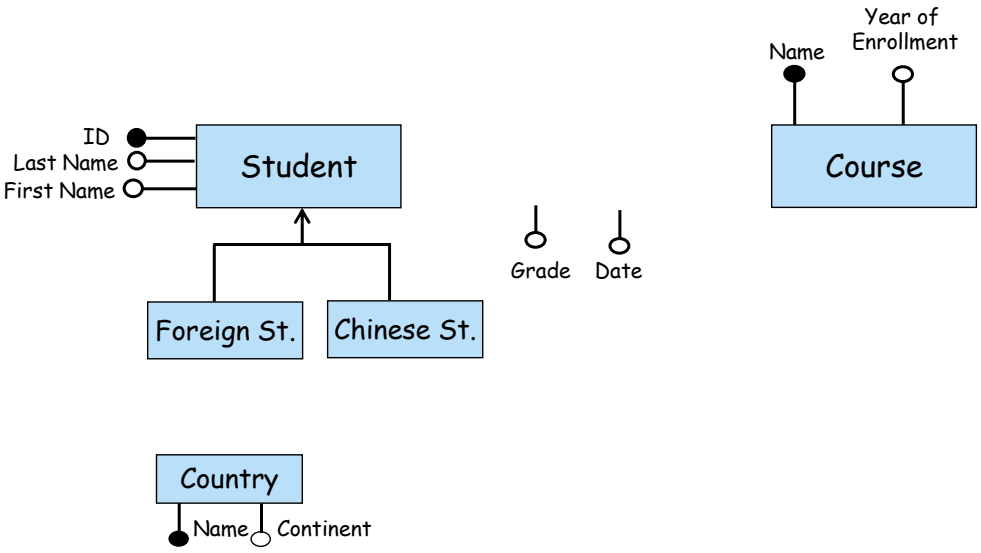
The bottom up strategy proceeds identifying first *the most elementary constructs*, which in the ER model are the attributes. Attributes are elementary in the sense that they are properties of other constructs, such as entities and relationships, and do not have properties themselves. We can examine carefully requirements and highlight items that we recognize as attributes, leading to the selection in the figure below. We also represent a first draft schema made only of attributes.



Bottom up strategy: identify attributes

In a second step, we may identify higher order constructs, such as entities. This process justifies the name *bottom up*, from elementary constructs to higher order constructs. This second selection leads to the following choices and schema.

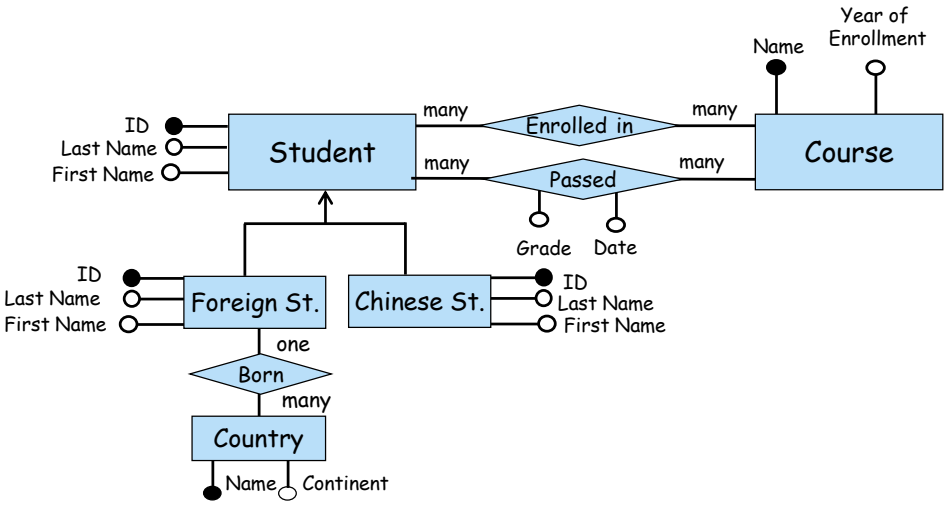
We are in China. Students are of two types: Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses. Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Course with Grade and Date.



Bottom up strategy: find entities

Notice that in the above schema we have added the generalization among *Student* and *Chinese + Foreign Student*. We have applied the inheritance property, assigning to Student common attributes to Foreign and Chinese Student. We see here a drawback of the bottom up strategy, that, using a metaphor, when you are in a forest makes you see first the leaves, then branches and only at the end the trees. Finally we may identify relationships in the requirements, and add them to the schema, connecting them to related attributes.

We are in China. Students are of two types, Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses, Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Course with Grade and Date.

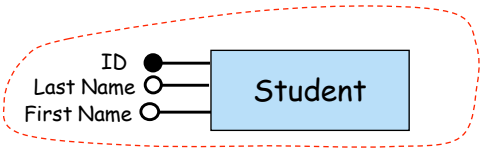


Bottom up strategy: identify relationships

Inside out

In the inside out strategy we try to avoid problems existing in the bottom up strategy; this is done identifying first *the most important concept*, or else one of the most important concepts in requirements, and modeling such concept with an entity. The importance can be established by the frequency of the term and related terms in requirements. In our case, one of the most important concepts is certainly Student, that we may modeled together with its attributes.

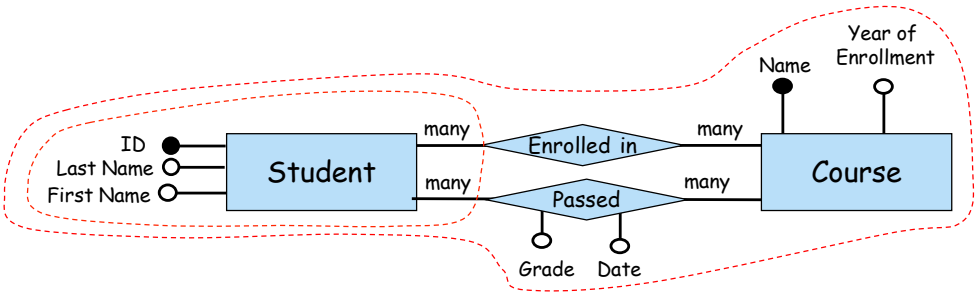
We are in China. Students are of two types, Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses, Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Course with Grade and Date.



Inside out strategy: first the most important entity

At this point we may move from Student toward the part of the schema referring to courses, see next figure.

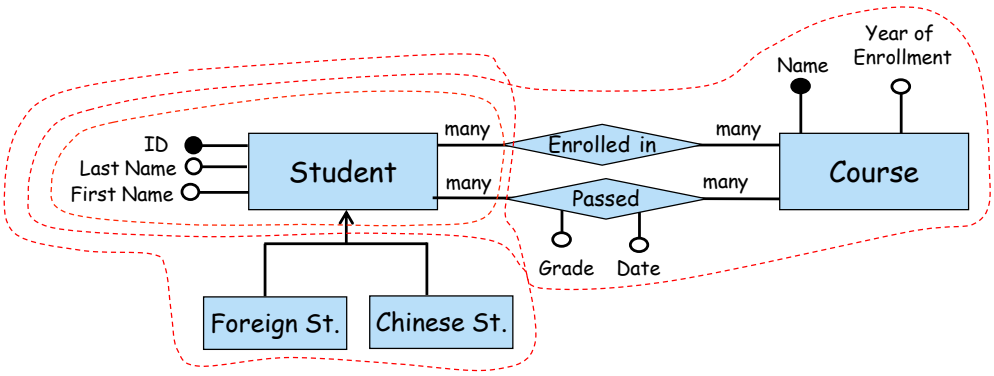
We are in China. Students are of two types, Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses. Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Course with Grade and Date.



Inside out strategy: now concepts that are close to the most important concept

Then we can move toward the two types of Student, foreign and chinese students.

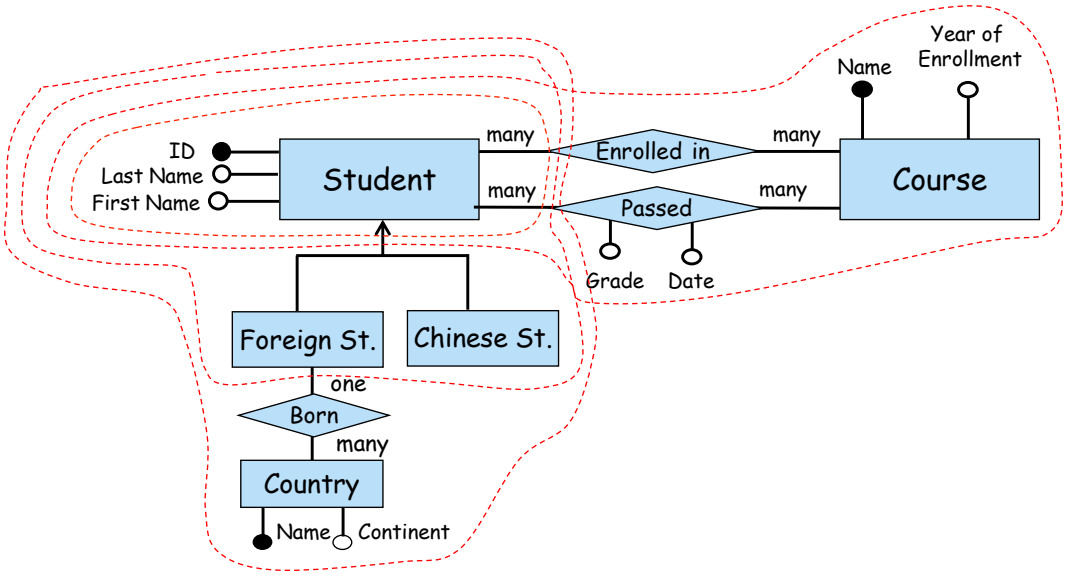
We are in China. Students are of two types, Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses. Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Course with Grade and Date.



Inside out strategy: now, move in the schema….

Finally, we cover all the requirements moving toward the rest of the schema (this movement from inside (Student) to outside (the constructs close to Student and then the constructs that are far from Student) gives the name to the strategy.

We are in China. Students are of two types, Foreign Students and Chinese Students. For both of them we want to represent ID, First Name and Last Name. Furthermore, for Foreign Students we want to represent the Country where they are born with Name and Continent. Students are related to courses. Courses have Name and Year of Enrollment. Students are enrolled in courses; furthermore they pass Course with Grade and Date.

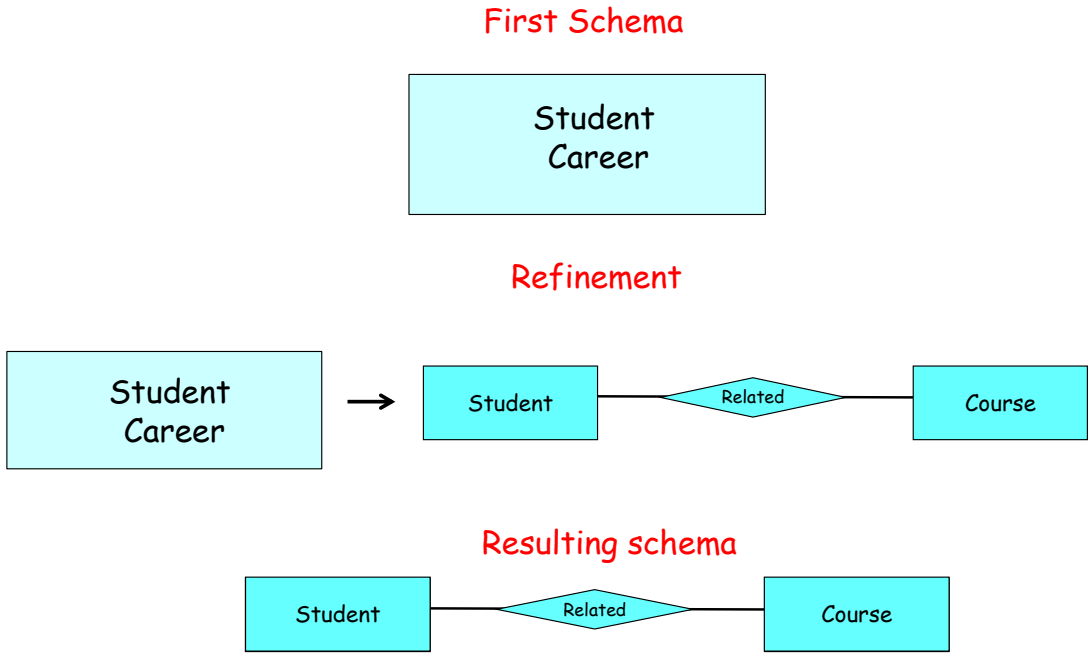


Inside out strategy: a different direction of navigation

Notice that the inside out strategy does not avoid the potential need of restructurings in the schema, but certainly, contrary to what happens in the case of the bottom up strategy, achieves early in the design process a complete view of the most important part of the schema.

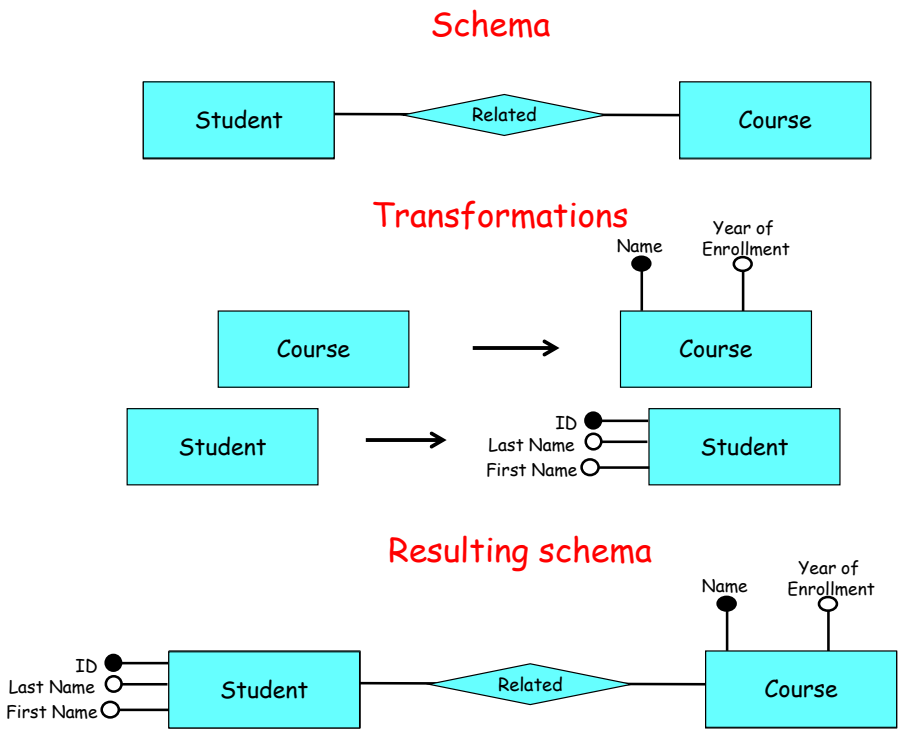
Top down

In the top down strategy we follow a design process that is exactly the opposite of the bottom up strategy. In a sense, in the top down strategy up to the beginning of the design process we model the whole schema. It is modeled initially with a unique entity, and then refining the entity with transformations that lead to a larger schema; such larger schema again represents the whole schema, with more constructs than before. See in the next figure the initial schema, with the first refinement and the resulting schema in the figure.



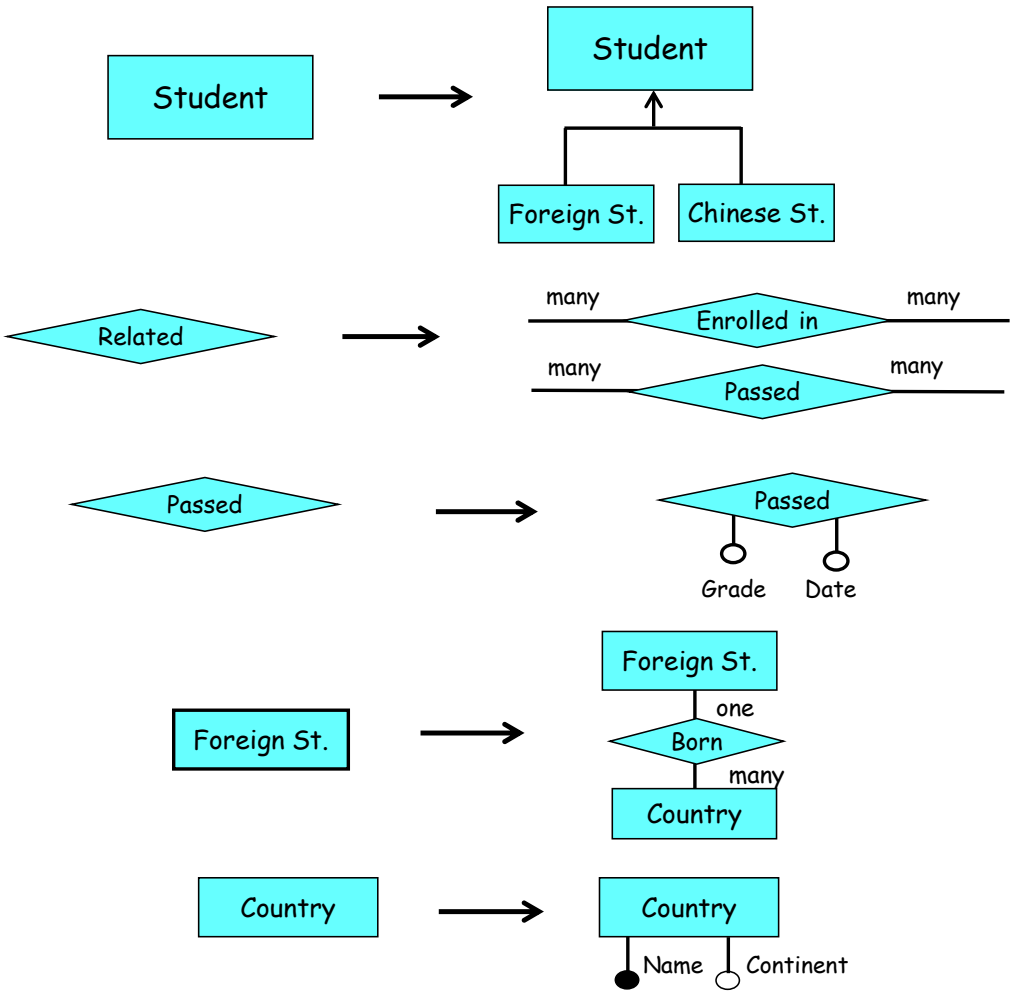
A first refinement obtained with the top down strategy

Notice that the name "related" associated to the relationship between Student and Course is an abstract name, that we acknowledge we shall later refine in more concrete relationships. After the above refinement, we can apply two refinements to the entities Student and Course, obtaining the following schema.



Second top down refinement

The following figure shows a possible choice of further refinements whose application leads to the final schema.



Further refinements

The top down strategy ends when all refinements have been applied, and the schema exactly corresponds to requirements. See in the Power point animation in the video lesson the top down strategy exemplified with ovals that tend to grow, as long as new refinements are applied.

In the top down strategy we proceed with transformations that respond to a unique strategy, namely to transform an abstract concept into a group of more concrete ones, so to be able to represent at each step the whole set of requirements, at a more and more concrete level.

Notice that in the top down strategy we minimize the risk of reorganizations on the schema. E.g. when we refine Student into Foreign Student + Chinese Student, we do not need to restructure the schema, as the inheritance property automatically assigns the properties of the parent entity Student to the child entities Foreign Student and Chinese Student.

Part 4 – Lesson 5 – An exercise on strategies

Exercise 4.2 – Look at the following requirements.

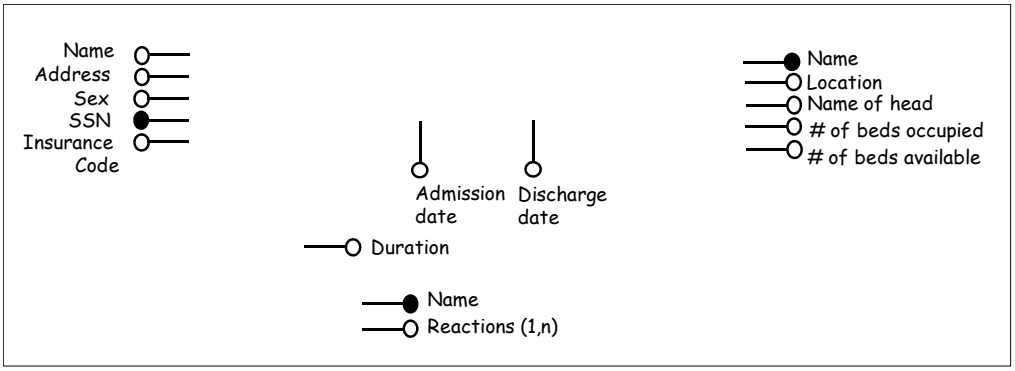
The hospital database stores data about patients, their admission and discharge from hospital's departments, and their treatments. For each patient, we know the name, address, sex, social security number, and insurance code. For each department, we know the department's name, its location, the name of the doctor who heads it, the number of beds available, and the number of beds occupied. Each patient goes possibly through multiple treatments during hospitalization; for each treatment, we store its name, duration, and the possible reactions to it the patient may have.

Adopt the three design strategies discussed in the previous lessons, so to experiment on the job their effectiveness.

Solution of Exercise 4.2

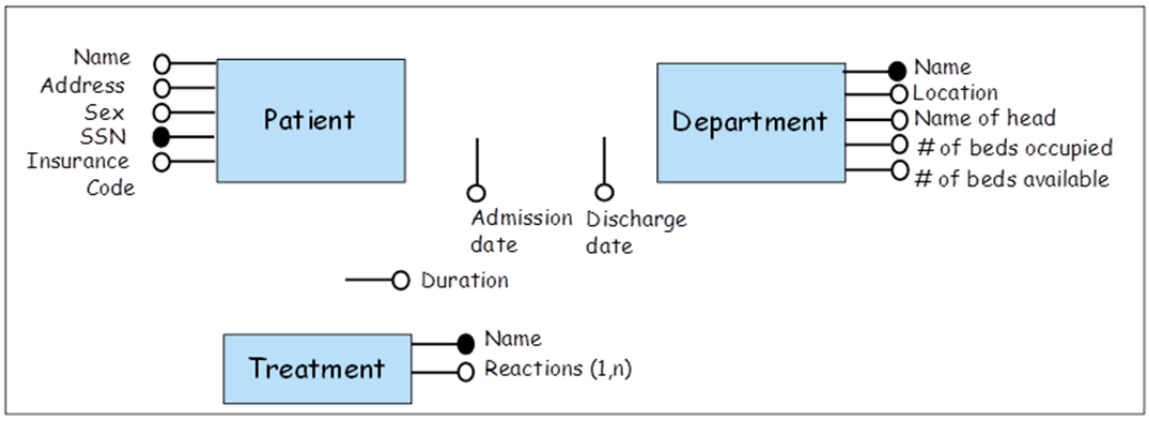
Bottom up strategy

We may initially select elementary properties in requirements, that will be modeled as attributes. Then we may cluster them according to the common entity or relationship that we envisage.



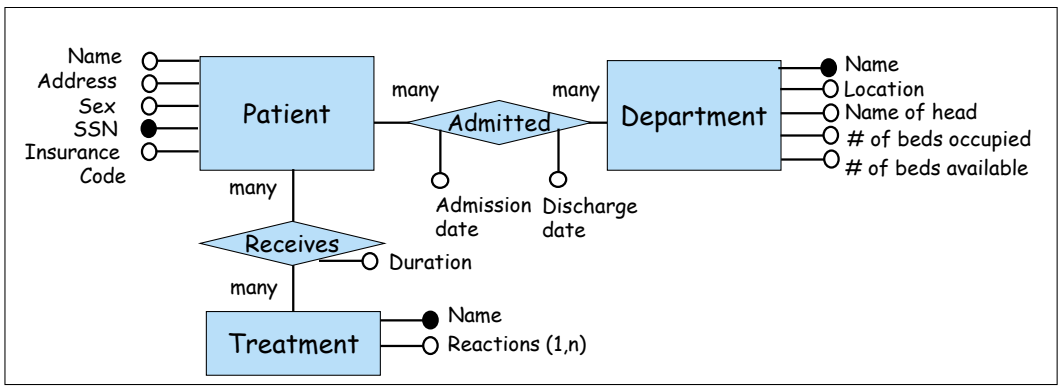
Initial schema in the bottom-up strategy

Now we can add first entities.



Added entities

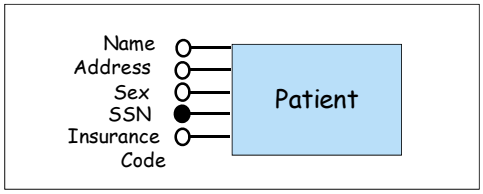
and then relationships.



Added relationships

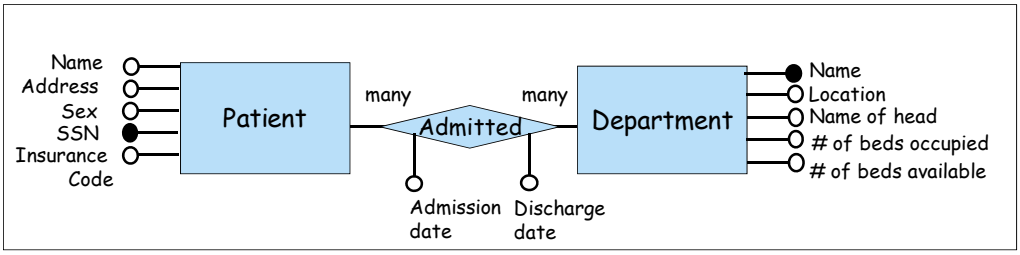
Inside out strategy

We can start from Patient, choosing its attributes.



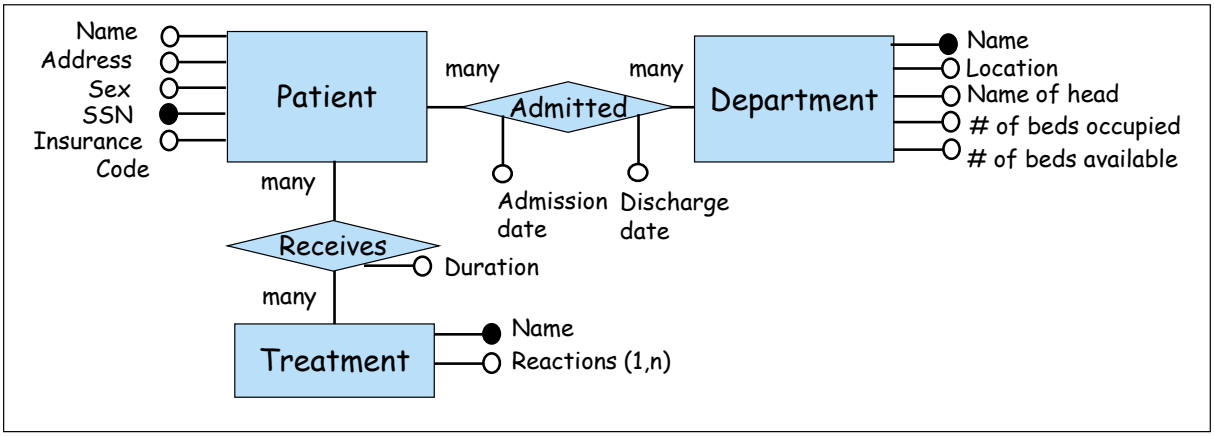
Initial schema in the inside-out strategy

Then we may move first toward Department, connected to Student with the relationship Admitted.



Schema extension toward Department

Then we may move from the other side to Treatment, connecting Patient to Treatment with the relationship “receives”.



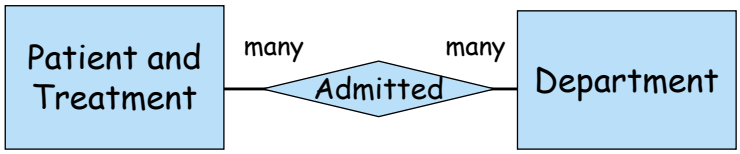
Schema extension toward Treatment

Top down strategy

We can start with a schema made of the entity *Patient therapies*.

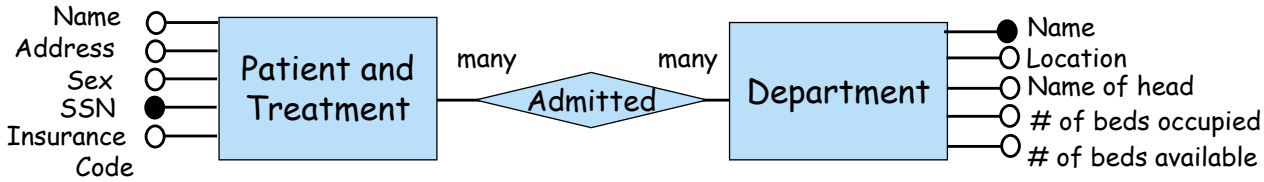
Starting schema in the top down strategy

Then we can refine with a schema with entities *Patient and Treatments* and *Department*, connected by relationship *Admitted*.

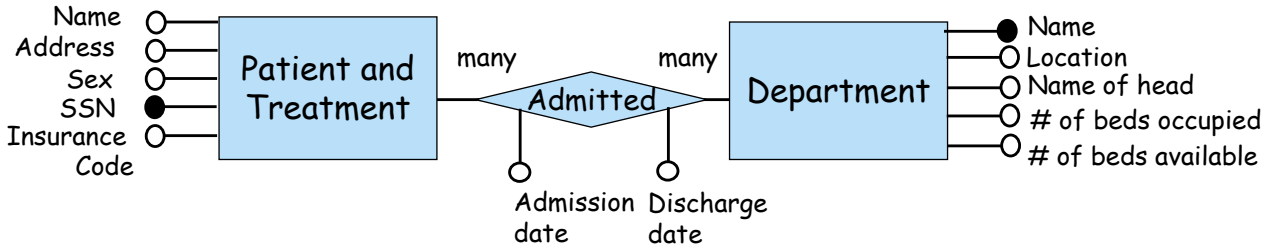


First refinement

At this point we can add attributes to entities Patient and Department, and subsequently to relationship Admitted.

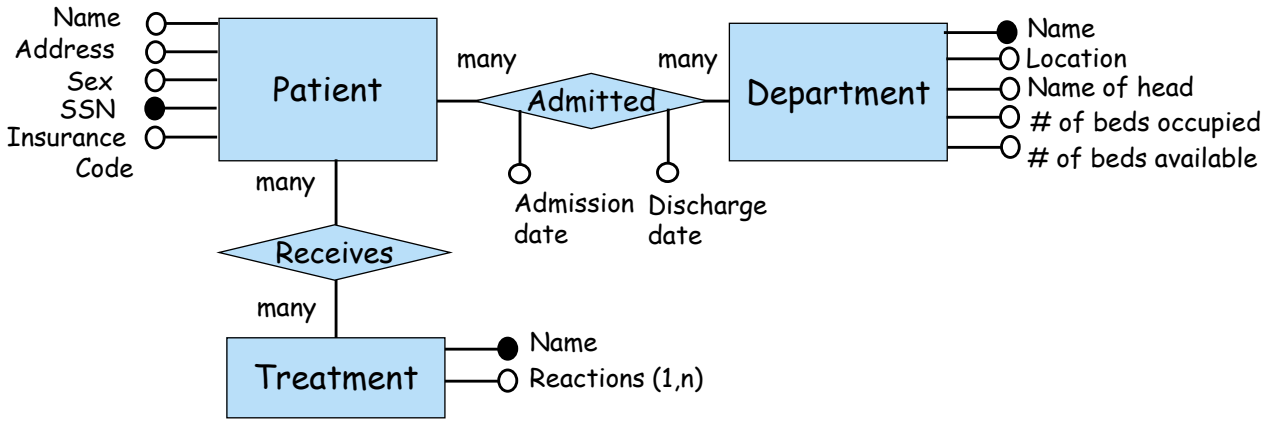


Second refinement on attributes of entities



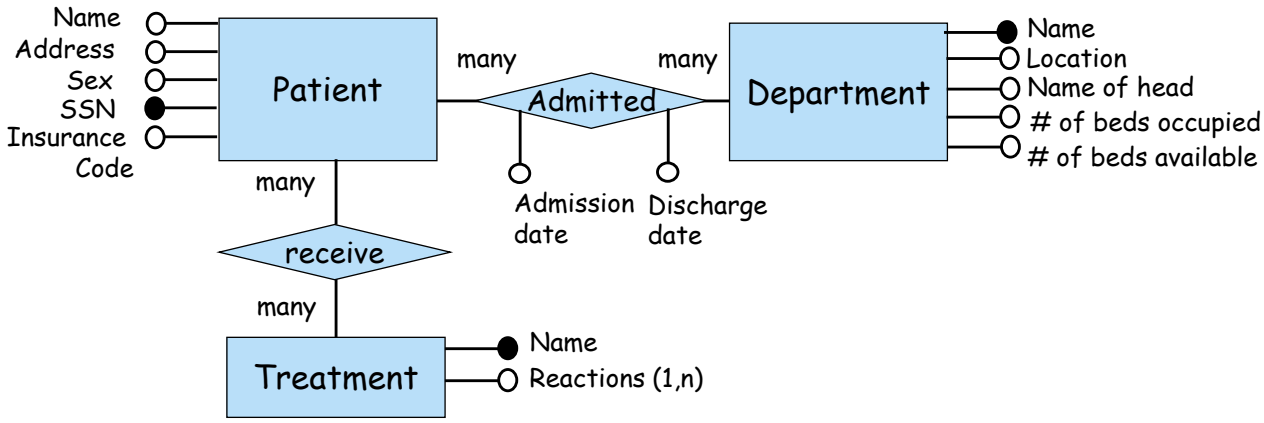
Third refinement on attributes of relationship *Admitted*

Then we can refine the other branch of the schema, leading to a relationship *Receives* between *Patient* and *Treatment*.



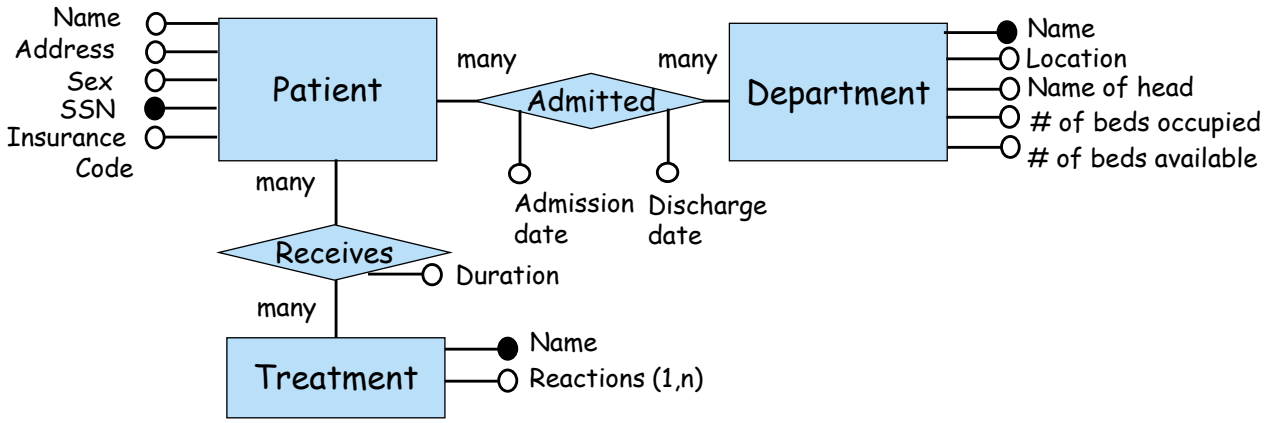
Fourth refinement creates a new entity and relationship

At this point we refine entity *Treatment*.



Fifth refinement on attributes of entity *Treatment*

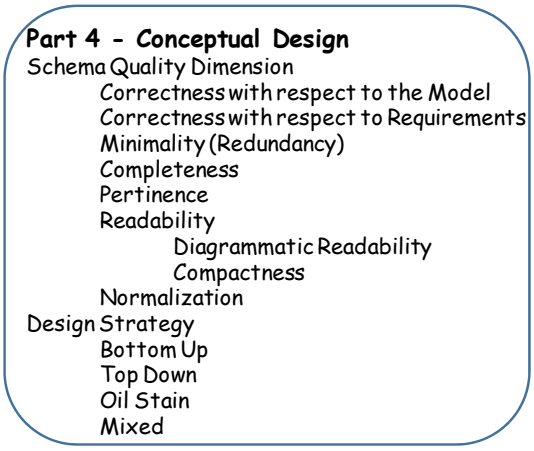
We finish refining relationship *Receives*.



Sixth refinement on attribute of Receives

The lesson ends here.

Concepts defined in Part 4



Part 4 – Exercise assignment

Solve exercises from 6.1 to 6.10 of Chapter 6 of Atzeni's book. Then compare your solutions with solutions provided in the course site.